MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2022/2023

ECE3166 ADVANCED MICROPROCESSORS

(All Sections / Groups)

5 JULY 2023
9:00 A.M. – 11:00 A.M.
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of 11 pages with 4 Questions only.
2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please print all your answers in the Answer Booklet provided.
4. The summary of instruction set and the formats of some relevant descriptors and table entry are provided in Appendices A, B, C and D, respectively.

Question 1

(a) Complex Instruction Set Computer (CISC) and Reduce Instruction Set Computer (RISC) are two most commonly used computer architecture in industry.

(i) Provide three characteristics of CISC and three characteristics of RISC, respectively.

[6 marks]

(ii) Give one example of a microprocessor architecture for CISC and another example for RISC.

[2 marks]

(iii) Give two types of microprocessor benchmark to compare the performance of microprocessor, and describe each of them.

[4 marks]

(b) Re-sketch the diagram in Figure Q1 on your answer sheet and label each part to show the software architecture of an 8086 microprocessor.

[13 marks]

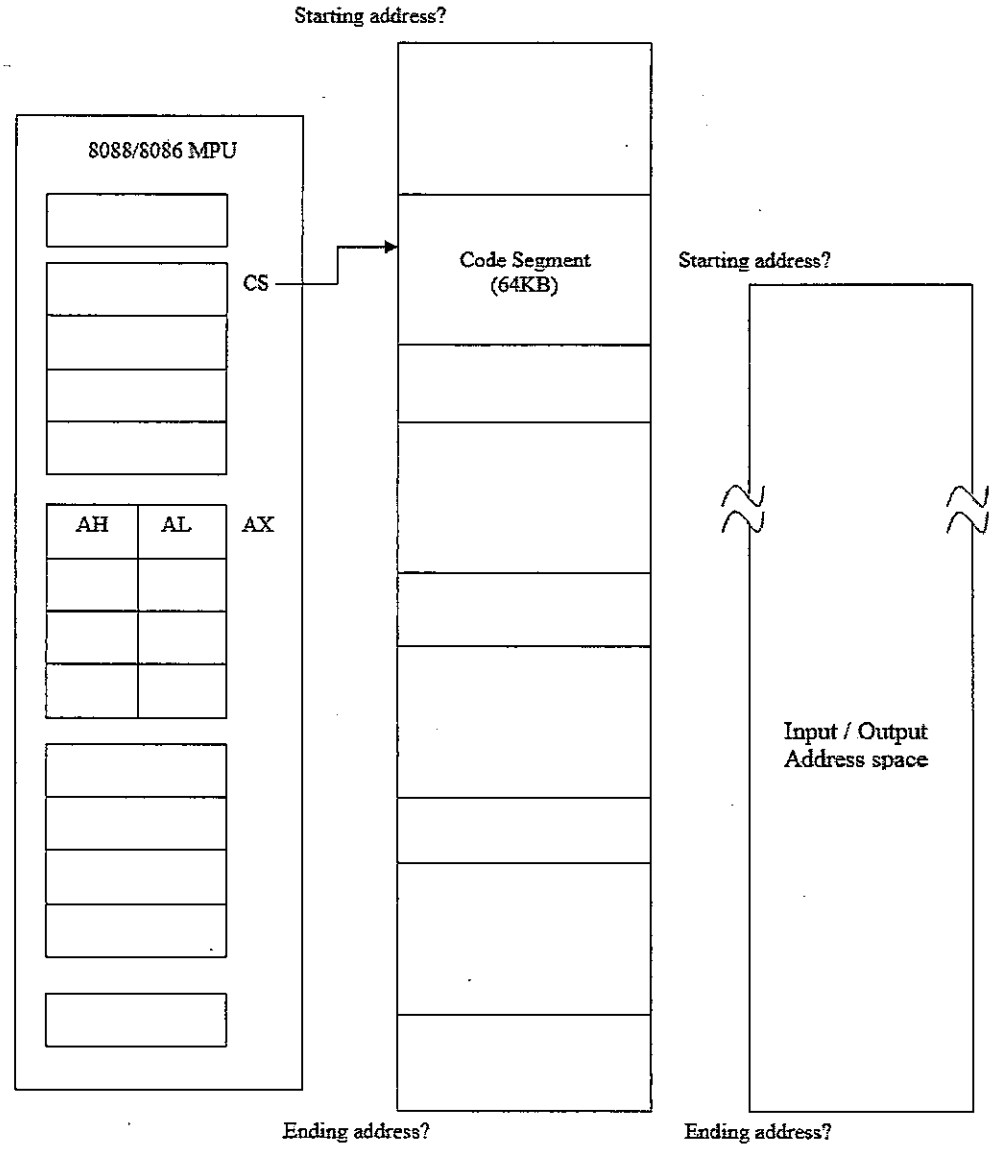


Figure Q1

Continued ...

Question 2

(a) Find the unknown value for the logical addresses corresponding to each of the following physical addresses. Show all steps of your calculations.

(i) B300H : 0334H = ?

[2 marks]

(ii) ? : 1156H = 12056H

[2 marks]

(iii) D900H : ? = D2234H

[2 marks]

(b) Table Q2 shows the contents of various registers and memory in an 80386DX microprocessor. Identify the **addressing mode**, compute the **physical address** (real-mode programming) of the destination operand for each of the following instructions, and state the **final updated content** (register / memory) after executing each of the instruction below. Show the steps of your calculations.
Note: Each of the following instruction is stand-alone and not related. Given CF=0.

(i) MOV FS:[DI] + 52H, AH

[4 marks]

(ii) ADD ES:[BX] + 80H, AX

[4 marks]

(iii) SUB BYTE PTR[DI], CH

[4 marks]

Table Q2

Register	Contents (Hex)
AX	1234H
BX	0200H
CX	10FFH
DX	2010H
SI	0040H
DI	0030H

Register	Contents (Hex)
CS	0300H
DS	0800H
ES	1000H
FS	1020H
GS	2000H
SS	5100H

Memory address (Hex)	Content (Hex)
10280	63
10281	E8
10282	5B
10283	A9
08030	88
08031	99

(c) A subroutine program is written to compare 22H to all the 100 bytes of content starting with address 1234H. As long as they are equal, the loop continue until all the bytes of content finishes. Complete the following subroutine to perform such tasks, and give a brief description on the function of CX and ZF in this subroutine program.

[7 marks]

```
                MOV   DL, ?
                MOV   SI, ?
                MOV   CX, ?
NEXT_BYTE:      ?     SI
                CMP    ?     ?
                LOOPE NEXT_BYTE
```

Continued ...

Question 3

(a) A Global Descriptor Table (GDT) is to be initialized with a base address of 02003300H. The table must be able to store 2048 descriptors.

(i) Determine the value to be loaded into the Global Descriptor Table Register (GDTR).

[3 marks]

(ii) If a value of 1600H is loaded into the Local Descriptor Table Register (LDTR), determine the address range of the Local Descriptor Table (LDT) descriptor.

[3 marks]

(b) The DS selector register is loaded with a value of 0223H

(i) Determine if the segment descriptor is located in GDT or LDT.

[2 marks]

(ii) Determine the Requested Privilege Level (RPL).

[1 mark]

(iii) Given that the base address of GDT and LDT are 10102000H and 12004000H, respectively, determine the address range of the segment descriptor.

[4 marks]

(iv) If the contents of the physical memory are:

[10102220H] = 4FFFH	[12004220H] = 2FFFH
[10102222H] = 3000H	[12004222H] = 1200H
[10102224H] = F300H	[12004224H] = AF10H
[10102226H] = 0501H	[12004226H] = 2200H

Determine the base address and the end address of the segment.

[4 marks]

(c) Assume that 32-bit paging is enabled on an IA-32 processor with a page size of 4kB. Given that the linear address of an operand is 22000820H, while the contents of the corresponding page directory entry and page table entry are 16008007H and 17308005H, respectively.

(i) Compute the physical address range of the page table and page frame.

[4 marks]

(ii) Determine the physical address range of the page directory entry if the base address of the page directory table is 0012 1000H.

[4 marks]

Continued ...

Question 4

(a) Given that the contents of the MMX registers are:

[MM0] = 2230 00A0 10F5 A032 H
[MM1] = 00F5 10B0 F0A2 0033 H
[MM2] = 9010 0345 10A0 0010 H

(i) Using the content of MM2 register, determine the values of the individual elements for 4×16 data.

[2 marks]

(ii) Evaluate the contents of MM0, MM1, and MM2 after the following sequence of instructions are executed:

```
PACKUSWB  MM0, MM2
PADDUSB   MM1, MM0
PCMPGTB   MM2, MM1
```

Note: Show all necessary working procedures. [10 marks]

(b) Listing Q4(b) shows manipulation of single precision floating point numbers using SSE instructions.

```
.DATA
OP_1   DD   18.0, 24.25, 1.0, 6.0
OP_2   DD   50.0, 2.0, 55.75, 3.0
OP_3   DD   38.2, 44.23, 55.0, 1010.25

.CODE
MAIN PROC
MOVUPS   XMM0, OWORD PTR OP_1
MOVUPS   XMM1, OWORD PTR OP_2
MOVUPS   XMM2, OWORD PTR OP_3
MULPS    XMM1, XMM0
MOVHLPS  XMM2, XMM1
RET
MAIN ENDP
END MAIN
```

Listing Q4(b)

Fill up Table Q4(b) to show the affected register and its content after execution of the following assembly lines. Show the contents in **hexadecimal**.

Table Q4(b)

Instruction	Affected register = Content
`MOVUPS XMM0, OWORD PTR OP_1`	`XMM0 = 40C00000 3F800000 41C20000 41900000`
`MOVUPS XMM1, OWORD PTR OP_2`	`XMM1 = 40400000 425F0000 40000000 42480000`
`MOVUPS XMM2, OWORD PTR OP_3`	`XMM2 = 447C9000 425C0000 4230EB85 4218CCCD`
`MULPS XMM1, XMM0`	
`MOVHLPS XMM2, XMM1`	

[13 marks]

Continued ...

Appendix A

Instruction Set Summary for Intel 8086/80186/80286/80386/80486

AAA - Ascii Adjust for Addition
AAS - Ascii Adjust for Subtraction
ADC - Add With Carry
ADD - Arithmetic Addition
AND - Logical And
BSF - Bit Scan Forward (386+)
BSR - Bit Scan Reverse (386+)
BSWAP - Byte Swap (486+)
BT - Bit Test (386+)
BTC - Bit Test with Compliment (386+)
BTR - Bit Test with Reset (386+)
BTS - Bit Test and Set (386+)
CALL - Procedure Call
CBW - Convert Byte to Word
CDQ - Convert Double to Quad (386+)
CLC - Clear Carry
CLD - Clear Direction Flag
CLI - Clear Interrupt Flag (disable)
CLTS - Clear Task Switched Flag (286+ privileged)
CMC - Complement Carry Flag
CMP - Compare
CMPS - Compare String (Byte, Word or Doubleword)
CMPXCHG - Compare and Exchange
CWD - Convert Word to Doubleword
CWDE - Convert Word to Extended Doubleword (386+)
DAA - Decimal Adjust for Addition
DAS - Decimal Adjust for Subtraction
DEC - Decrement
DIV - Divide
ENTER - Make Stack Frame (80188+)
ESC - Escape
HLT - Halt CPU
IDIV - Signed Integer Division
IMUL - Signed Multiply
IN - Input Byte or Word From Port
INC - Increment
INT - Interrupt
INTO - Interrupt on Overflow
INVD - Invalidate Cache (486+)
INVLPG - Invalidate Translation Look-Aside Buffer Entry (486+)
IRET/IRETD - Interrupt Return
Jxx - Jump Instructions Table
JCXZ/JECXZ - Jump if Register (E)CX is Zero
JMP - Unconditional Jump
LAHF - Load Register AH From Flags
LAR - Load Access Rights (286+ protected)
LDS - Load Pointer Using DS
LEA - Load Effective Address
LES - Load Pointer Using ES
LFS - Load Pointer Using FS (386+)
LGDT - Load Global Descriptor Table (286+ privileged)
LIDT - Load Interrupt Descriptor Table (286+ privileged)
LGS - Load Pointer Using GS (386+)
LLDT - Load Local Descriptor Table (286+ privileged)
LMSW - Load Machine Status Word (286+ privileged)

Continued ...

LOCK - Lock Bus
LODS - Load String (Byte, Word or Double)
LOOP - Decrement CX and Loop if CX Not Zero
LOOPE/LOOPZ - Loop While Equal / Loop While Zero
LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal
LSL - Load Segment Limit (286+ protected)
LSS - Load Pointer Using SS (386+)
LTR - Load Task Register (286+ privileged)
MOV - Move Byte or Word
MOVS - Move String (Byte or Word)
MOVSX - Move with Sign Extend (386+)
MOVZX - Move with Zero Extend (386+)
MUL - Unsigned Multiply
NEG - Two's Complement Negation
NOP - No Operation (90h)
OR - Inclusive Logical OR
OUT - Output Data to Port
POP - Pop Word off Stack
POPA/POPAD - Pop All Registers onto Stack (80188+)
POPF/POPFD - Pop Flags off Stack
PUSH - Push Word onto Stack
PUSHA/PUSHAD - Push All Registers onto Stack (80188+)
PUSHF/PUSHFD - Push Flags onto Stack
RCL - Rotate Through Carry Left
RCR - Rotate Through Carry Right
REP - Repeat String Operation
REPE/REPZ - Repeat Equal / Repeat Zero
REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero
RET/RETF - Return From Procedure
ROL - Rotate Left
ROR - Rotate Right
SAHF - Store AH Register into FLAGS
SAL/SHL - Shift Arithmetic Left / Shift Logical Left
SAR - Shift Arithmetic Right
SBB - Subtract with Borrow/Carry
SCAS - Scan String (Byte, Word or Doubleword)
SETS - Set if Signed (386+)
SETNS - Set if Not Signed (386+)
SETC - Set if Carry (386+)
SETNC - Set if Not Carry (386+)
SETO - Set if Overflow (386+)
SETP/SETPE - Set if Parity / Set if Parity Even (386+)
SETNP/SETPO - Set if No Parity / Set if Parity Odd (386+)
SGDT - Store Global Descriptor Table (286+ privileged)
SIDT - Store Interrupt Descriptor Table (286+ privileged)
SHL - Shift Logical Left
SHR - Shift Logical Right
SHLD/SHRD - Double Precision Shift (386+)
SLDT - Store Local Descriptor Table (286+ privileged)
SMSW - Store Machine Status Word (286+ privileged)
STC - Set Carry
STD - Set Direction Flag
STI - Set Interrupt Flag (Enable Interrupts)
STOS - Store String (Byte, Word or Doubleword)
STR - Store Task Register (286+ privileged)
SUB - Subtract
TEST - Test for Bit Pattern
WBINVD - Write-Back and Invalidate Cache (486+)
XCHG - Exchange
XOR - Exclusive OR

Continued ...

Mnemonic	Meaning	Format	Operation
LOOP	Loop	LOOP Short-label	(CX) ← (CX) − 1 Jump is initiated to location defined by short-label if (CX) ≠ 0; otherwise, execute next sequential instruction
LOOPE/LOOPZ	Loop while equal/ loop while zero	LOOPE/LOOPZ Short-label	(CX) ← (CX) − 1 Jump to location defined by short-label if (CX) ≠ 0 and (ZF) = 1; otherwise, execute next sequential instruction
LOOPNE/ LOOPNZ	Loop while not equal/ loop while not zero	LOOPNE/LOOPNZ Short-label	(CX) ← (CX) − 1 Jump to location defined by short-label if (CX) ≠ 0 and (ZF) = 0; otherwise, execute next sequential instruction

Mnemonic	Meaning	Format	Operation	Flags Affected
MOVS	Move string	MOVSB/MOVSW	((ES)0 + (DI)) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None
CMPS	Compare string	CMPSB/CMPSW	Set flags as per ((DS)0 + (SI)) − ((ES)0 + (DI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
SCAS	Scan string	SCASB/SCASW	Set flags as per (AL or AX) − ((ES)0 + (DI)) (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
LODS	Load string	LODSB/LODSW	(AL or AX) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2	None
STOS	Store string	STOSB/STOSW	((ES)0 + (DI)) ← (AL or AX) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None

Prefix	Used with:	Meaning
REP	MOVS STOS	Repeat while not end of string CX ≠ 0
REPE/REPZ	CMPS SCAS	Repeat while not end of string and strings are equal CX ≠ 0 and ZF = 1
REPNE/REPNZ	CMPS SCAS	Repeat while not end of string and strings are not equal CX ≠ 0 and ZF = 0

Mnemonic	Meaning	Format	Operation	Flags Affected
CLD	Clear DF	CLD	(DF) ← 0	DF
STD	Set DF	STD	(DF) ← 1	DF

Continued ...

Appendix B

MMX Instruction Set Summary

Mnemonic	Name
MOVQ	Move quad word
MOVD	Move double word
PADDB	Packed add bytes
PADDW	Packed add words
PADDD	Packed add double words
PADDSB	Packed add signed bytes
PADDSW	Packed add signed words
PADDUSB	Packed add unsigned bytes
PADDUSW	Packed add unsigned words
PSUBB	Packed subtract bytes
PSUBW	Packed subtract words
PSUBD	Packed subtract double words
PSUBSB	Packed subtract signed bytes
PSUBSW	Packed subtract signed words
PSUBSD	Packed subtract signed double words
PSUBUSB	Packed subtract unsigned bytes
PSUBUSW	Packed subtract unsigned words
PMULLW	Packed multiply low words
PMULHW	Packed multiply high words
PMADDWD	Packed multiply and add words
PSRAW	Packed shift right arithmetic words
PSRAD	Packed shift right arithmetic double words
PCMPEQB	Packed compare for equal bytes
PCMPEQW	Packed compare for equal words
PCMPEQD	Packed compare for equal double words
PCMPGTB	Packed compare for greater than bytes
PCMPGTW	Packed compare for greater than words
PCMPGTD	Packed compare for greater than double words
PAND	Bitwise logical AND
PANDN	Bitwise logical AND NOT
POR	Bitwise logical OR
PXOR	Bitwise logical exclusive-OR
PSLLW	Packed shift left logical word
PSLLD	Packed shift left logical double word
PSLLQ	Packed shift left logical quad word
PSRLW	Packed shift right logical word
PSRLD	Packed shift right logical double word
PSRLQ	Packed shift right logical quad word
PACKUSWB	Pack words to bytes with unsigned saturation
PACKSSWB	Pack words to bytes with signed saturation
PACKSSDW	Pack double words to words with signed saturation
PUNPCKHBW	Unpack high packed data bytes to words
PUNPCKHWD	Unpack high packed data words to double words
PUNPCKHDQ	Unpack high packed data double words to quad word
PUNPCKLBW	Unpack low packed data bytes to words
PUNPCKLWD	Unpack low packed data words to double words
PUNPCKLDQ	Unpack low packed data double words to quad word

Continued ...

Appendix C

Summary of Selected FPU Instructions

Mnemonic	Description
FINIT	Initialize floating-point unit
FLD	Load floating-point value onto FPU register stack
FSTP	Store floating-point value and pop the FPU register stack
FADD	Add floating-point values
FSUB	Subtract floating-point values
FMUL	Multiply floating-point values
FDIV	Divide floating-point values

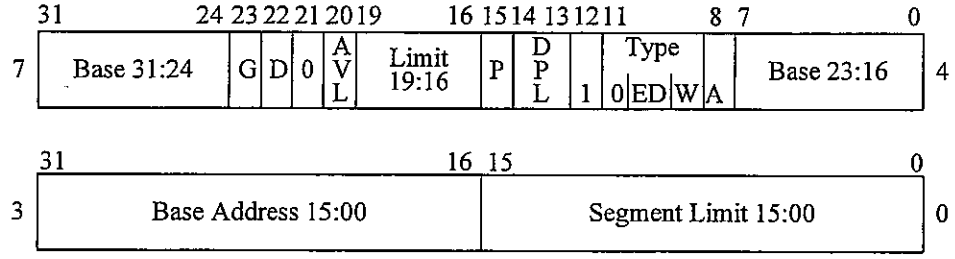
Summary of Selected SSE Instructions

Mnemonic	Description
MOVAPS	Move aligned packed single-precision floating-point values
MOVUPS	Move unaligned packed single-precision, floating-point
MOVHPS	Move high packed single-precision floating-point
MOVHLPS	Move packed single-precision floating-point high to low
MOVLHPS	Move packed single-precision floating-point low to high
MOVLPS	Move low packed single-precision floating-point
MOVMSKPS	Move packed single-precision floating-point mask
MOVSS	Move scalar single-precision floating-point
ADDSS	Scalar add pair of operands
ADDPS	Packed add pairs of operands
SUBSS	Scalar subtract pair of operands
SUBPS	Packed subtract pairs of operands
MULSS	Scalar multiply pair of operands
MULPS	Packed multiply pairs of operands
DIVSS	Scalar divide pair of operands
DIVPS	Packed divide pairs of operands

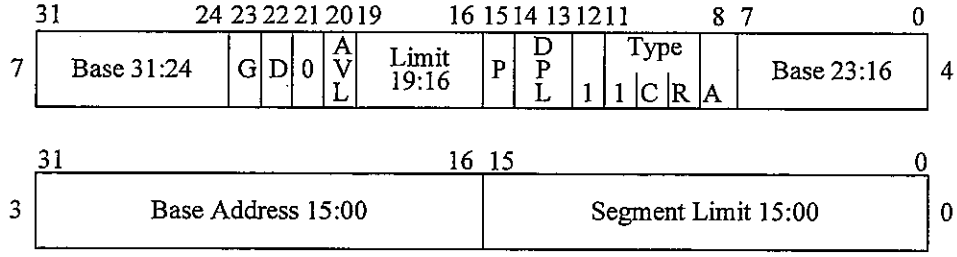
Continued ...

Appendix D

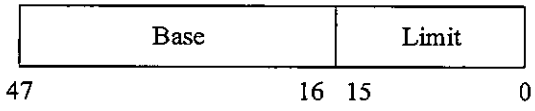
Data-Segment Descriptor:



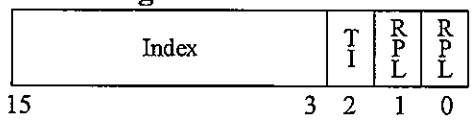
Code-Segment Descriptor:



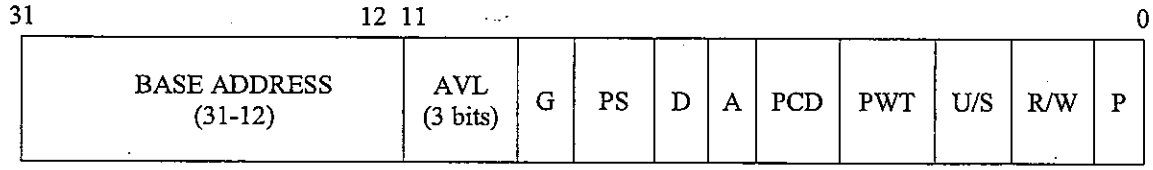
Global Descriptor Table Register (GDTR) and Interrupt Descriptor Table Register (IDTR):



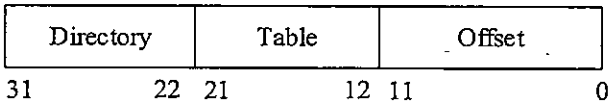
Selector Register:



Page Directory or Page Table Entry:



Linear Address:



End of Page